AGGREGATION OPERATORS

Recall: Aggregation Operations are Column Operations!

- An aggregation operator may not appear in the WHERE clause unless it is in a subquery contained in a HAVING clause or a SELECT list, and the column being aggregated is an outer reference.

```
SELECT empID, SUM(amount)
FROM Sales
GROUP BY Employee
HAVING SUM(amount) > 20000;
```

vs.

```
SELECT empID, SUM(amount)
FROM Sales
GROUP BY Employee
WHERE empID IN (
    SELECT MAX(empID)
    FROM Employees);
```


A FINAL EXAMPLE

```
SELECT bar, SUM(qty) AS sumQ
FROM Sells
GROUP BY bar
HAVING SUM(qty) > 4;
```

Sells:

bar	beer	price	qty
Joe	Bud	3.00	2
Joe	Miller	4.00	2
Tom	Bud	3.50	1
Tom	Miller	4.25	4
Jane	Bud	3.25	1
Jane	Miller	4.75	3
Jane	Coors	4.00	2

Result:

bar	sumQ
Tom	5
Jane	6

CROSS PRODUCT

- A.K.A. Cartesian Product. Denoted by: X
- Evaluating joins involves combining two or more relations.
- Given two relations, *S* and *R*, each row of *S* is paired with each row of *R*.
- Resulting Schema: one attribute from each attribute of *S* and *R*.

EXAMPLE: CROSS PRODUCT

Sells:

bar	beer	price
Joe	Bud	3.00
Tom	Miller	4.00
Jane	Lite	3.25

Frequents:

drinker	bar
Aaron	Joe
Mary	Jane

Sells X Frequents

(bar)	beer	price	drinker	(bar)
Joe	Bud	3.00	Aaron	Joe
Joe	Bud	3.00	Mary	Jane
Tom	Miller	4.00	Aaron	Joe
Tom	Miller	4.00	Mary	Jane
Jane	Lite	3.25	Aaron	Joe
Jane	Lite	3.25	Mary	Jane

Result:

drinker
Aaron

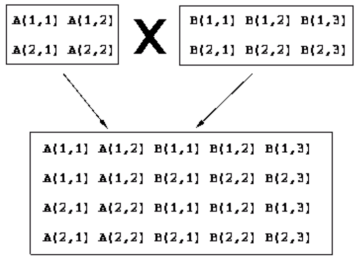
```
SELECT drinker
FROM Frequents, Sells
WHERE beer = 'Bud' AND Frequents.bar = Sells.bar;
```

CROSS PRODUCT

- In general:

	Table A	Table B	A X B
Rows	N	M	N * M
Columns	C	K	C + K

Example: Cross product of 2x2-table **A** and 2x3-table **B**.



CARTESIAN DIVISION

- Division is the opposite of the Cartesian Product.

  e.g. Table A is the Cartesian product of Table B X Table C.

Table A:

C	D
4	3
4	1
4	7
8	3
8	1
8	7

Table B:

C
4
8

Table A / Table C

Table C:

D
3
1
7

Table A / Table B

JOINED RELATIONS

- **Join operations** take two relations and returns as a result another relation.
- A join operation is a Cartesian product which requires that tuples in the two relations match (under some condition). It also specifies the attributes that are present in the result of the join.

EXAMPLE: JOIN OPERATIONS

Course

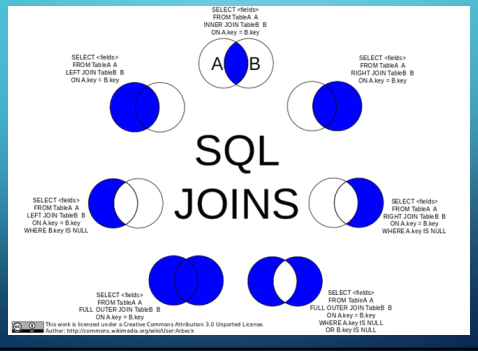
course_id	title	dept_name	credits
BIO-301	Genetics	Biology	4
CS-190	Game Design	Comp Sci	4
CS-315	Robotics	Comp Sci	3

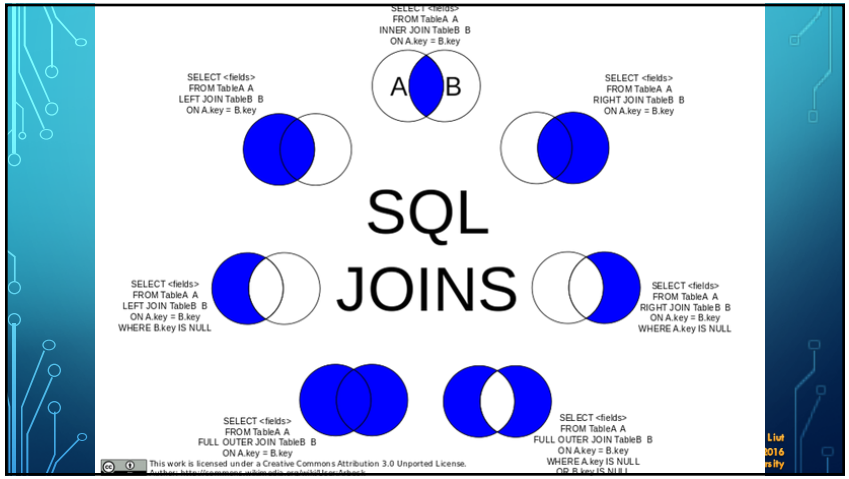
Prereq

course_id	prereq_id
BIO-301	BIO-101
CS-190	CS-101
CS-347	CS-101

Observe:
1. The prerequisite information is missing for CS-315.
2. The course information is missing for CS-347.

TYPES OF JOINS





OUTER JOIN

- An extension of the join operation that avoids loss of information.
- Suppose you have two relations *R* and *S*. A tuple of *R* that has no tuple of *S* with which it joins is said to be *dangling*.
  - Similarly for a tuple of *S*.
- Computes the join and then adds tuples from one relation that does not match tuples in the other relation to the result of the join.
- Outerjoin preserves dangling tuples by padding them with NULL.

LEFT OUTER JOIN

Course:

course_id	title	dept_name	credits
BIO-301	Genetics	Biology	4
CS-190	Game Design	Comp Sci	4
CS-315	Robotics	Comp Sci	3

Prereq:

course_id	prereq_id
BIO-301	BIO-101
CS-190	CS-101
CS-347	CS-101

Result:

course_id	title	dept_name	credits	prereq_id
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp Sci	4	CS-101
CS-315	Robotics	Comp Sci	3	NULL

RIGHT OUTER JOIN

Course:

course_id	title	dept_name	credits
BIO-301	Genetics	Biology	4
CS-190	Game Design	Comp Sci	4
CS-315	Robotics	Comp Sci	3

Prereq:

course_id	prereq_id
BIO-301	BIO-101
CS-190	CS-101
CS-347	CS-101

Result:

course_id	title	dept_name	credits	prereq_id
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp Sci	4	CS-101
CS-347	NULL	NULL	NULL	CS-101

FULL OUTER JOIN

Course:

course_id	title	dept_name	credits
BIO-301	Genetics	Biology	4
CS-190	Game Design	Comp Sci	4
CS-315	Robotics	Comp Sci	3

Prereq:

course_id	prereq_id
BIO-301	BIO-101
CS-190	CS-101
CS-347	CS-101

Result:

course_id	title	dept_name	credits	prereq_id
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp Sci	4	CS-101
CS-315	Robotics	Comp Sci	3	NULL
CS-347	NULL	NULL	NULL	CS-101

INNER JOIN

Course:

course_id	title	dept_name	credits
BIO-301	Genetics	Biology	4
CS-190	Game Design	Comp Sci	4
CS-315	Robotics	Comp Sci	3

Prereq:

course_id	prereq_id
BIO-301	BIO-101
CS-190	CS-101
CS-347	CS-101

Course **INNER JOIN** Prereq **ON**
Course.course_id = Prereq.course_id

Result:

course_id	title	dept_name	credits	prereq_id
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp Sci	4	CS-101

OUTER JOINS

- *R* OUTER JOIN *S* is the core of an outer join expression.
- Modified by the following:
  1. Optional NATURAL in front of OUTER.
     - Check equality on all common attributes.
     - No two attributes with the same name in the output.
  2. Optional ON <condition> after JOIN.
  3. Optional LEFT, RIGHT, or FULL in front of OUTER.
     - LEFT = pad dangling tuples of *R* only.
     - RIGHT = pad dangling tuples of *S* only.
     - FULL = pad both; this is the DEFAULT choice.

EXAMPLE: OUTER JOIN

NOTE: (1,2) joins with (2,3), but the other two tuples are dangling

R:

A	B
1	2
4	5

S:

B	C
2	3
6	7

R **NATURAL FULL OUTER JOIN** S

Result:

A	B	C
1	2	3
4	5	NULL
NULL	6	7